Milad Mikeal

CS162

9/23/18

Project 1

Design:

- Call menu function
    - Ask user to start program or quit
    - Validate the user's input
    - Exit if user opted to quit
- Have user input rows, columns, steps, and starting point
    - Validate each input
- Create ant object
    - Constructor
        - Allocate memory for 2D board array
        - Specify orientation
        - Print board at its current state
    - Steps function
        - Method to keep track of how many steps have been taken so far
    - Move function
        - Method to determine ant's movement each turn
        - Update location and orientation
        - Increment number of steps taken
        - Print board
    - Print function
        - Method to print the board at its current state each turn
    - Free memory
        - Method to free memory of dynamically allocated 2D boards array
- Call ant's move function until number of steps taken by ant so far is equal to number of steps specified by user
- Call ant method to free board space

Menu Test Table:

Test Case	**Input**	**Expected Outcome**	**Outcome**
Input too low	Input < 1	Prompt user to re-enter value	User prompted to re-enter value
Input too high	Input > 2	Prompt user to re-enter value	User prompted to re-enter value
Input in range	Input = 1 or Input = 2	Proceed accordingly depending on user's input	Program proceeded accordingly
Non-numeric input	Input = alpha	Prompt user to re-enter value	User prompted to re-enter value

User Inputs: Rows test table

Test Case	**Input**	**Expected Outcome**	**Outcome**
Input too low	Input < 1	Prompt user to re-enter value	User prompted to re-enter value
Input too high	Input > 50	Prompt user to re-enter value	User prompted to re-enter value
Input in range	0 < Input < 50	Proceed accordingly depending on user's input	Program proceeded accordingly
Non-numeric input	Input = alpha	Prompt user to re-enter value	User prompted to re-enter value

User Inputs: Columns test table

Test Case	**Input**	**Expected Outcome**	**Outcome**
Input too low	Input < 1	Prompt user to re-enter value	User prompted to re-enter value
Input too high	Input > 50	Prompt user to re-enter value	User prompted to re-enter value
Input in range	0 < Input < 50	Proceed accordingly depending on user's input	Program proceeded accordingly
Non-numeric input	Input = alpha	Prompt user to re-enter value	User prompted to re-enter value

User Inputs: Steps test table

Test Case	**Input**	**Expected Outcome**	**Outcome**
Input too low	Input < 1	Prompt user to re-enter value	User prompted to re-enter value
Input too high	Input > INT_MAX	Prompt user to re-enter value	User prompted to re-enter value
Input in range	0 < Input < INT_MAX	Proceed accordingly depending on user's input	Program proceeded accordingly
Non-numeric input	Input = alpha	Prompt user to re-enter value	User prompted to re-enter value

User Inputs: Starting Row test table

Test Case	**Input**	**Expected Outcome**	**Outcome**
Input too low	Input < 1	Prompt user to re-enter value	User prompted to re-enter value
Input too high	Input > rows	Prompt user to re-enter value	User prompted to re-enter value
Input in range	0 < Input < rows	Proceed accordingly depending on user's input	Program proceeded accordingly
Non-numeric input	Input = alpha	Prompt user to re-enter value	User prompted to re-enter value

User Inputs: Starting Column test table

Test Case	**Input**	**Expected Outcome**	**Outcome**
Input too low	Input < 1	Prompt user to re-enter value	User prompted to re-enter value
Input too high	Input > columns	Prompt user to re-enter value	User prompted to re-enter value
Input in range	0 < Input < columns	Proceed accordingly depending on user's input	Program proceeded accordingly
Non-numeric input	Input = alpha	Prompt user to re-enter value	User prompted to re-enter value

Reflection:

I honestly did not think this assignment was as difficult as it was made out to be. It was pretty straight-forward in terms of what was required of us. Being thrown at a large project like this was a little daunting at first, but once you actually begin, you realize it's actually not too bad.

I did make a few slight changes. The most notable change I made involved the location. Originally, during the design plan, I was going to make the location an object itself. However, as I continued with the design plan, I realized I was making the assignment more difficult than it needed to be. So I adjusted the design plan.

Once I completed the design plan, translating it to code was pretty easy, for the most part. The toughest part was the Ant object's makeMove() method. I needed to find a way to represent each position on the board as a white space, a black space, and a space with ant. After trying out several ways to do this, I finally found a sound solution. I used numbers to represent each space on the board: 1 – white space, 2 – black space, 3 – white space with an ant, and 4 – black space with an ant. I needed to differentiate the ant space into black or white so that it could change accordingly to a black space or a white space after each turn. When it came down to my printBoard() ant method, I used the numbers to print the appropriate symbol representing each space, " " for white, # for black, and * for white space with ant and black space with ant. Once I figured that part out, the rest was pretty easy.

There were other changes I made, but it was mostly just cleaning up. For example, I made a function called userInputs() to prompt the user to input values for rows, columns, steps, startingRow, and startingCol to keep my main function clean and neat. I passed the variables by reference. I then made another function called toRandomize() to ask the user if they would like to randomize the starting point. If they opted to randomize the starting point, I made another function called userInputs2() for the user to input only the rows, columns, and steps. I used an if/else statement to determine which userInputs() function would be called. I also put several parts of the main function in a while loop so that I could ask the user if they would like to re-run the program once the initial turn was done. I created another function called runAgain() to ask the user if they would like to re-run the program. If they opted not to, the while loop would end, as would the program.

Essentially, I made a lot of cleaning up changes that just made the main function cleaner. Putting each function in its own file just made everything neater and more organized. The functions were called accordingly. All in all, I actually really enjoyed this project, and I felt like I learned a lot. I really enjoyed the freedom the project gave me.